UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/479,363	01/07/2000	Timothy James Graser	RO999-122	2954

24038 7590 08/18/2004

MARTIN & ASSOCIATES, LLC
P O BOX 548
CARTHAGE, MO 64836-0548

EXAMINER	
LY, ANH	
ART UNIT	**PAPER NUMBER**
2172	

DATE MAILED: 08/18/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/479,363	Applicant(s) GRASER, TIMOTHY JAMES	
	Examiner Anh Ly	Art Unit 2172	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *21 May 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-23* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-23* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is response to Applicant's response filed on 5/21/2004.

2. Claims 1, 3-8, 10-15 and 17-23 are pending in this application.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

Claims1, 6, 12,m 13, 20 and 21 are rejected under the judicially created doctrine of double patenting over claims 1, 2, 5, 7 and 13 of U. S. Patent No. 5,943,497 since the claims, if allowed, would improperly extend the "right to exclude" already granted in the patent.

The subject matter claimed in the instant application is fully disclosed in the patent and is covered by the patent since the patent and the application are claiming common subject matter, as follows: configuration data and replacement class.

Furthermore, there is no apparent reason why applicant was prevented from presenting claims corresponding to those of the instant application during prosecution of the application, which matured into a patent. See *In re Schneller*, 397 F.2d 350, 158 USPQ 210 (CCPA 1968). See also MPEP § 804.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to

> be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 3-8, 10-15 and 17-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 5,943,497 issued to Bohrer et al. (hereinafter Bohrer) in view of US Patent No. 6,405,209 issued to Obendorf.

With respect to claim 1, Bohrer teaches at least one processor (fig. 1, item 110, col. 5, lines 22-23);

a memory coupled to the at least one processor (fig. 1, item 120, col. 5, lines 22-23);

class configuration data comprising a plurality of entries residing in the memory, each class configuration entry including a key-value pair, wherein the key includes information relating to a selected processing context and the value includes configuration data for a class in the selected processing context (see fig. 5 and fig. 6, the key value pair here is the factory class and configuration data and class and the processing of the context of the class, col. 4, lines 50-59 and col. 9, lines 32-62) wherein the key comprises context information appended to a class identifier (container ID in this case: col. 8, lines 9-15); and

an object oriented class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data (instance of class: col. 3, lines 5-10 and col. 7, lines 34-38) to access the appropriate entry in the class configuration data (see figs 5 and 6; also see abstract and col. 7, lines 15-21).

Bohrer teaches allowing a new configuration data to replace existing configuration data within an existing object-oriented without changing the source code of the existing OO program, and factory object is created and the configuration data generally contains the class tokens utilized by their corresponding factories and persistent container defines a scope of the class or object being created with a particular physical system and server process within the system (see figs 2 and 3, col. 6, lines 57-67 and col. 7, lines 1-54). Bohrer does not clearly teaches updating or replacing the data store as a factory object in a relational database or storing object in a relational database to be instantiated.

However, Obendorf teaches an apparatus for instantiating and initializing an object from a relational database. As shown in FIG. 1 is an exemplary hardware that has *at least one processor; a memory coupled to the at least one processor.* As shown in FIG. 3B is a reference table as *class configuration data comprising a plurality of entries residing in the memory, each class configuration entry including a key-value pair, wherein the key includes* TableName object ID as *information relating* to the process of creating the table object as *a selected processing context and the value includes* the class I D as *configuration data for a class in the selected processing context.* Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding

to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37)

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Bohrer with the teachings of Obendorf by incorporating the use of datastore as factory class for updating it in a relational database and storing objects in a relational database to be instantiated. The motivation being to have allowed the use of datastore storing in a relational database for easing to update and manipulate in the object-oriented for controlling configuration of object creation.

With respect to claim 3, Bohrer teaches wherein the class identifier comprises a class token that comprises a text string (class token: col. 7, lines 34-38 and col. 9, lines 35-40; also see fig. 4, item 302).

With respect to claim 4, Bohrer teaches a factory object that generates an instance of the selected class by accessing the appropriate entry in the class configuration data using the key (col. 4, lines 50-58 and col. 10, lines 10-28).

With respect to claim 5, Bohrer teaches a key generator mechanism that generates the key from a class identifier and from the context information (see fig. 5 for context of class information; see abstract, col. 4, lines 1-10; also see col. 6, lines 57-67 and col. 7, lines 1-21).

With respect to claim 6, Bohrer teaches retrieving configuration data corresponding to the class in a selected processing context using a

corresponding key that includes information relating to the selected processing context. wherein the key comprises context information appended to a class identifier (see fig. 5 and fig. 6, col. 98, lines 15-32); and

instantiating the instance of the class using the retrieved configuration data. (col. 3, lines 5-10 and col. 7, lines 34-38; also see col. 9, lines 15-32).

Bohrer teaches allowing a new configuration data to replace existing configuration data within an existing object-oriented without changing the source code of the existing OO program, and factory object is created and the configuration data generally contains the class tokens utilized by their corresponding factories and persistent container defines a scope of the class or object being created with a particular physical system and server process within the system (see figs 2 and 3, col. 6, lines 57-67 and col. 7, lines 1-54). Bohrer does not clearly teaches updating or replacing the data store as a factory object in a relational database or storing object in a relational database to be instantiated.

However, Obendorf teaches instantiating and initializing an object from a relational database. As disclosed by Obendorf, if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question (Obendorf, Col. 5, lines 20-37). As seen, a ClassID 218 as *configuration data is retrieved* to pass to the creation call

CoCreateInstance(), which locates the class factory for the object associated with the ClassID in table 240 as the step of *instantiating the instance of the class using the retrieved configuration data.* Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37). In other words, the technique as discussed above performed *an object oriented! class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data,* and Obendorf further discloses *the, key comprises context information appended to a class identifier* (Fig. 3B).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Bohrer with the teachings of Obendorf by incorporating the use of datastore as factory class for updating it in a relational database and storing objects in a relational database to be instantiated. The motivation being to have allowed the use of datastore storing in a relational database for easing to update and manipulate in the object-oriented for controlling configuration of object creation.

With respect to claim 7, Bohrer teaches storing the configuration data with the corresponding key (col. 5, lines 42-55 and col. 7, lines 55-67 and col. 8, lines 1-5).

With respect to claim 8, Bohrer teaches storing the configuration data with the corresponding key comprises the step of generating a key from a class identifier and from the context information (col. 6, lines 57-67 and col. 7, lines 1-21).

With respect to claim 10, Bohrer teaches wherein the class identifier comprises a class token that comprises a text string (col. 7, lines 34-38 and col. 9, lines 35-40; also see fig. 4, item 302).

With respect to claim 11, Bohrer teaches generating the key from a class identifier and from the context information (see fig. 5 for context of class information; see abstract, col. 4, lines 1-10; also see col. 6, lines 57-67 and col. 7, lines 1-21).

With respect to claim 12, Bohrer teaches generating a key that comprises information relating to a current processing context appended to a class identifier for the existing class (container ID in this class: col. 8, lines 9-15);

storing configuration data for the existing class using the key (see fig. 5, col. 4, lines 50-59);

replacing the configuration data for the existing class with configuration data for the replacement class while maintaining the same key (col. 7, lines 15-21);

initiating the creation of an instance of the replacement class (col. 3, lines 5-10 and col. 7, lines 15-21);

retrieving the configuration data for the replacement class using the generated key (col. 9, lines 15-32); and

creating an instance of the replacement class according to the retrieved configuration data for the replacement class (col. 7, lines 10-40).

Bohrer teaches allowing a new configuration data to replace existing configuration data within an existing object-oriented without changing the source code of the existing OO program, and factory object is created and the configuration data generally contains the class tokens utilized by their corresponding factories and persistent container defines a scope of the class or object being created with a particular physical system and server process within the system (see figs 2 and 3, col. 6, lines 57-67 and col. 7, lines 1-54). Bohrer does not clearly teaches updating or replacing the data store as a factory object in a relational database or storing object in a relational database to be instantiated.

However, Obendorf teaches instantiating and initializing an object from a relational database. As disclosed by Obendorf, if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question (Obendorf, Col. 5, lines 20-37). As

seen, a ClassID 218 as *configuration data is retrieved* to pass to the creation call CoCreateInstance(), which locates the class factory for the object associated with the ClassID in table 240 as the step of *instantiating the instance of the class using the retrieved configuration data.* Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37). In other words, the technique as discussed above performed *an object oriented! class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data,* and Obendorf further discloses *the, key comprises context information appended to a class identifier* (Fig. 3B).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Bohrer with the teachings of Obendorf by incorporating the use of datastore as factory class for updating it in a relational database and storing objects in a relational database to be instantiated. The motivation being to have allowed the use of datastore

storing in a relational database for easing to update and manipulate in the object-oriented for controlling configuration of object creation.

With respect to claim 13, Bohrer teaches an object oriented class replacement mechanism that generates an instance of a selected class by using a key that includes information relating to a selected processing context to access an appropriate entry in class configuration data stored external to the class, wherein the key comprises context information appended to a class identifier and signal bearing media bearing the object oriented class replacement mechanism (see figs 5 & 6 for factory object and context information of class: col. 8, lines 35-54 and col. 7, lines 15-22).

Bohrer teaches allowing a new configuration data to replace existing configuration data within an existing object-oriented without changing the source code of the existing OO program, and factory object is created and the configuration data generally contains the class tokens utilized by their corresponding factories and persistent container defines a scope of the class or object being created with a particular physical system and server process within the system (see figs 2 and 3, col. 6, lines 57-67 and col. 7, lines 1-54). Bohrer does not clearly teaches updating or replacing the data store as a factory object in a relational database or storing object in a relational database to be instantiated.

However, Obendorf teaches instantiating and initializing an object from a relational database. As disclosed by Obendorf, if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to

the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question (Obendorf, Col. 5, lines 20-37). As seen, a ClassID 218 as *configuration data is retrieved* to pass to the creation call CoCreateInstance(), which locates the class factory for the object associated with the ClassID in table 240 as the step of *instantiating the instance of the class using the retrieved configuration data.* Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37). In other words, the technique as discussed above performed *an object oriented! class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data,* and Obendorf further discloses *the, key comprises context information appended to a class identifier* (figs 3A and 3B).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Bohrer with the teachings of Obendorf by incorporating the use of datastore as factory class for updating it in a relational database and storing objects in a relational database to be instantiated. The motivation being to have allowed the use of datastore storing in a relational database for easing to update and manipulate in the object-oriented for controlling configuration of object creation.

With respect to claims 14-15, Bohrer discloses wherein said signal bearing media comprises recordable media; wherein said signal bearing media comprises transmission media (storage device and floppy disks: col. 5, lines 42-57 and col. 6, lines 45-48);

Claim 17 is essentially the same as claim 3 except that it is directed to a program product rather than an apparatus, and is rejected for the same reason as applied to the claim 3 hereinabove.

Claim 18 is essentially the same as claim 4 except that it is directed to a program product rather than an apparatus, and is rejected for the same reason as applied to the claim 4hereinabove.

Claim 19 is essentially the same as claim 5 except that it is directed to a program product rather than an apparatus, and is rejected for the same reason as applied to the claim 5 hereinabove.

With respect to claim 20, Bohrer teaches class configuration data comprising a plurality of entries residing in the memory, each class configuration entry including a key-value pair, wherein the key includes information relating to

a selected processing context and the value includes configuration data for a class in the selected processing context (container ID in this case: col. 8, lines 9-15);

a key generator mechanism residing in the memory and executed by the at least one processor that generates the key from the class identifier and from the context information, wherein the key comprises the context information appended to a text string class identifier (instance of class: col. 3, lines 5-10 and col. 7, lines 34-38); and

an object oriented class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using the key to access the appropriate entry in the class configuration data, the class replacement mechanism comprising a factory object that generates an instance of the selected class by accessing the appropriate entry in the class configuration data using the key (see figs 5 and 6; also see abstract and col. 7, lines 15-21).

Bohrer teaches allowing a new configuration data to replace existing configuration data within an existing object-oriented without changing the source code of the existing OO program, and factory object is created and the configuration data generally contains the class tokens utilized by their corresponding factories and persistent container defines a scope of the class or object being created with a particular physical system and server process within the system (see figs 2 and 3, col. 6, lines 57-67 and col. 7, lines 1-54). Bohrer does not clearly teaches at least one processor, a memory coupled to the at least

one processor, and updating or replacing the data store as a factory object in a relational database or storing object in a relational database to be instantiated.

Obendorf teaches an apparatus for instantiating and initializing an object from a relational database. As shown in FIG. 1 is an exemplary hardware that has *at least one processor; a memory coupled to the at least one processor.* As shown in FIG. 3B is a reference table as *class configuration data comprising a plurality of entries residing in the memory, each class configuration entry including a key-value pair, wherein the key includes* TableName object ID as *information relating* to the process of creating the table object as *a selected processing context and the value includes* the class I D as *configuration data for a class in the selected processing context.* Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37). In other words, the technique as discussed above performed an object oriented! class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data, and Obendorf further discloses

the, key comprises context information appended to a class identifier (figs. 3A & 3B).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Bohrer with the teachings of Obendorf by incorporating the use of datastore as factory class for updating it in a relational database and storing objects in a relational database to be instantiated. The motivation being to have allowed the use of datastore storing in a relational database for easing to update and manipulate in the object-oriented for controlling configuration of object creation.

With respect to claim 21, Bohrer teaches an object oriented class replacement mechanism that generates an instance of a selected class by using a key that includes information relating to a selected processing context to access an appropriate entry in class configuration data stored external to the

class, wherein the key comprises context information appended to a text string class identifier, the class replacement mechanism comprising a factory object that generates an instance of the selected class by accessing the appropriate entry in the class configuration data using the key, the class replacement mechanism further comprising a key generator mechanism that generates the key from the text string class identifier and from the context information, and signal bearing media bearing the object oriented class replacement mechanism ((see figs 2 and 3, col. 6, lines 57-67 and col. 7, lines 1-54). Bohrer does not clearly teaches updating or replacing the data store as a

factory object in a relational database or storing object in a relational database to be instantiated.

However, Obendorf teaches instantiating and initializing an object from a relational database. As disclosed by Obendorf, if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question (Obendorf, Col. 5, lines 20-37). As seen, a ClassID 218 as *configuration data is retrieved* to pass to the creation call CoCreateInstance(), which locates the class factory for the object associated with the ClassID in table 240 as the step of *instantiating the instance of the class using the retrieved configuration data.* Obendorf discloses if the client requests object creation by the RDBMS 126, the client sends a ClassID 218 as an argument to the creation call CoCreateInstance(). CoCreateInstance locates the class factory for the object associated with the ClassID 218 in table 240, loads the class factory into memory, and invokes the constructor corresponding to the ClassID 218, which creates the object in question. As seen, an object as an instance of a selected class is created by using ClassID 218 and creation call CoCreateInstance(), class factory as context information to access the reference table (Col. 5, lines 20-37). In other words, the technique as discussed above, performed *an object oriented! class replacement mechanism residing in the memory and executed by the at least one processor that generates an instance*

of a selected class by using a key that includes context information to access the appropriate entry in the class configuration data, and Obendorf further discloses *the, key comprises context information appended to a class identifier* (figs 3A and 3B).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Bohrer with the teachings of Obendorf by incorporating the use of datastore as factory class for updating it in a relational database and storing objects in a relational database to be instantiated. The motivation being to have allowed the use of datastore storing in a relational database for easing to update and manipulate in the object-oriented for controlling configuration of object creation.

With respect to claims 22-23, Bohrer teaches wherein the signal bearing media comprises recordable media and wherein the signal bearing media comprises transmission media ((floppy disk and analog communication links: col. 6, lines 45-48).

Contact Information

6. Any inquiry concerning this communication should be directed to Anh Ly whose telephone number is (703) 306-4527 or E-Mail: **anh.ly@uspto.gov**. The examiner can be reached on Monday - Friday from 8:00 AM to 4:00 PM.

If attempts to reach the examiner are unsuccessful, see the examiner's supervisor, John Breene, can be reached on (703) 305-9790.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to: Central Fax Center (703) 872-9306

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal

Drive, Arlington, VA, Fourth Floor (receptionist).

Inquiries of a general nature or relating to the status of this application should be

directed to the Group receptionist whose telephone number is (703) 305-3900.

JEAN M. CORRIELUS
PRIMARY EXAMINER

ANH LY
AUG. 9th, 2004